目錄

前言 1.1
InfluxDB介绍 1.2
- 安装使用 1.2.1
- 名词解释 1.2.2

基本操作 1.3
- 增 1.3.1
- 删与改 1.3.2
- 查 1.3.3
- 数据库与表的操作 1.3.4
- 数据保存策略（Retention Policies） 1.3.5
- 连续查询（Continuous Queries） 1.3.6
- 用户管理 1.3.7

第三方库API接口 1.4
- NodeJS 1.4.1

数据展示工具 1.5
- Grafana简单使用 1.5.1

接下去可以看些什么？ 1.6

InfluxDB简明手册

手册不断完善中，欢迎 **pull request!**
如果对你有所帮助，请给一个**Star!**
项目地址: https://github.com/xtutu/influxdb-handbook

手册在线地址: https://www.gitbook.com/read/book/xtutu/influxdb-handbook

作者博客: http://www.xtutu.me/

项目介绍

写这本手册的时候**InfluxDB**版本为： **v0.10**

本手册只是简单的上手教程，并不是官方文档的详细翻译。 之所以不是官方文档的中文翻译，有两方面原因：

1. 精力有限，全部翻译量实在是太大了。
2. 官方文档是开源的：https://github.com/influxdata/docs.influxdata.com
   它所采用的文档工具gohugo.io是支持多语言设计的。
   所以有兴趣参与的同学，可以直接在Github上Fork。

使用方法

如果嫌在线速度慢的话，也直接在本地生成。具体方法如下：
1 安装NodeJS
2 安装gitbook

```
npm install gitbook -g
```

3 clone本仓库
4 生成html代码

```
gitbook build
```

然后就可以直接在本地看啦

InfluxDB介绍

InfluxDB用Go语言编写的一个开源分布式时序、事件和指标数据库，和传统是数据库相比有不少不同的地方。

类似的数据库有Elasticsearch、Graphite等。

特点

1. 提供了Http接口的API来操作数据
2. 提供了类似sql的数据库语句
3. 其它...

用途

一般用来储存实时数据，配合一套UI界面来展示信息。

安装使用

下载地址

直接官网下载就好，非常简单。
https://influxdata.com/downloads/#influxdb

安装

```
sudo dpkg -i influxdbName.deb
```

启动

```
sudo service influxdb start
```

使用

启动成功之后，我们就可以开始使用influxDB啦！

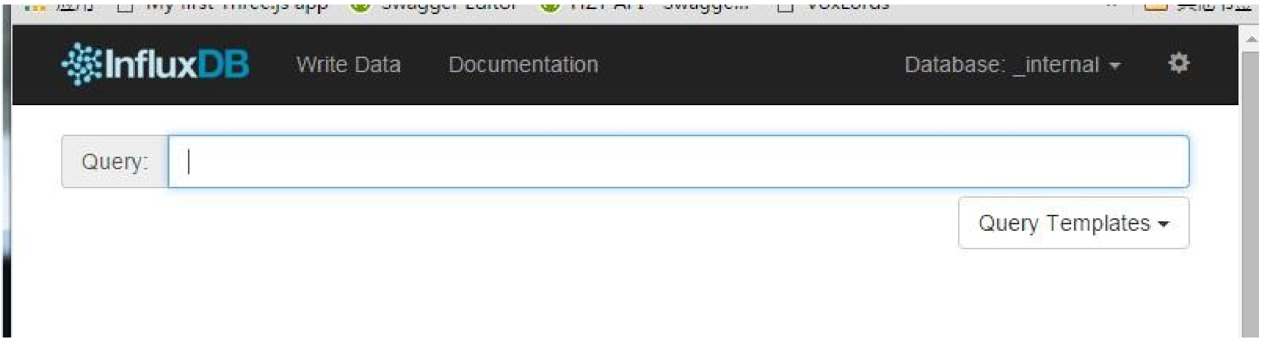
命令行

在命令行中直接输入influx，就可以管理数据库了。

```
root@xtutu:~# influx
Visit https://enterprise.influxdata.com to register for updates,
 InfluxDB server management, and monitoring.
Connected to http://localhost:8086 version 0.10.0
InfluxDB shell 0.10.0
> show databases
name: databases
---------------
name
_internal
mydb
```

使用web页面来操作

在浏览器中输入localhost:8083 即可进入web管理页面。



创建一个数据库

```
> CREATE DATABASE "testDB"
> show databases
name: databases
---------------
name
_internal
mydb
testDB
```

使用数据库

```
> use testDB
Using database testDB
```

现在我们就可以在这个数据库上进行各种操作了！

表的增删改查等操作在之后的几个章节里

名词解释

在上一个章节中，已经建立了一个名为testDB的数据库。在之后的演示中，都将在这个数据库上操作。

在InfluxDB中有不少名词，初学者非常容易搞混，这一节主要就是对这些名词进行解释。

场景定义

我们有一个数据库名为testDB，里面有一张表weather用于记录：多个地区在几组海拔下的一天的温度变化，所以表中有以下字段：

1. 时间 time
2. 温度 temperature
3. 湿度 humidity
4. 地区 area
5. 海拔 altitude

与传统数据库中的名词做比较

influxDB中的名词	传统数据库中的概念
database	数据库
measurement	数据库中的表
points	表里面的一行数据

InfluxDB中独有的一些念概

Point由时间戳（time）、数据（field）、标签（tags）组成。

Point属性	传统数据库中的概念
time	每个数据记录时间，是数据库中的主索引(会自动生成)
fields	各种记录值（没有索引的属性）也就是记录的值：温度， 湿度
tags	各种有索引的属性：地区，海拔

这里不得不提另一个名词：**series**：

所有在数据库中的数据，都需要通过图表来展示，而这个series表示这个表里面的数据，可以在图表上画成几条线：通过**tags**排列组合算出来。

比如有如下数据：

```
> select* from weather
name: weather
-------------
time                    altitude        area    humidity
temperature
1456386985094000000     1000            北      18
17
1456386985094000000     5000            上      20
47
1456386985094000000     5000            北      26
68
1456386985094000000     1000            广      17
83
1456387267668000000     1000            上      12
77
1456387267668000000     1000            北      16
20
1456387267668000000     5000            广      -3
66
1456387267668000000     5000            上      19
60
```

它的series为：

```
> show series from weather
name: weather
-------------
_key                              altitude        area
weather,altitude=1000,area=北     1000            北
weather,altitude=5000,area=北     5000            北
weather,altitude=5000,area=上     5000            上
weather,altitude=1000,area=广     1000            广
weather,altitude=1000,area=上     1000            上
weather,altitude=5000,area=广     5000            广
```

基本操作

本章将介绍InfluxDB中的一些基本操作，包括数据的增删改查、数据库与表的操作等。

增

在名词解释这一章节中，我们看到在weather中的有不少数据。
本节将演示下如何为数据库插入数据。

通过命令行

```
use testDB
insert weather,altitude=1000,area=北 temperature=11,humidity=-4
```

这样，我们就向数据库中添加了一条数据。

通过Http接口

InfluxDB提供了Http的API接口，所以我们也可以通过下面的方式来插入数据。

```
curl -i -XPOST 'http://localhost:8086/write?db=testDB' --data-bi
nary 'weather,altitude=1000,area=北 temperature=11,humidity=-4'
```

Line Protocol格式

插入数据的格式似乎比较奇怪，这是因为influxDB储存数据所采用的是Line Protocol格式。

在上面两个插入数据的方法中，都有一样的部分。

> weather,altitude=1000,area=北 temperature=11,humidity=-4

其中:

1. weather ： 表名
2. altitude=1000,area=北 ： tag
3. temperature=11,humidity=-4 ：field

具体的格式介绍可以看官方的文档

删与改

在InfluxDB中并没有提供数据的删除与修改方法。
不过我们可以通过数据保存策略（Retention Policies）来实现删除。
具体请看：数据保存策略（Retention Policies）这一章节。

查

本节将演示下查询数据的一些常用方法。

通过命令行

```
use testDB
# 查询最新的三条数据
SELECT * FROM weather ORDER BY time DESC LIMIT 3
```

通过Http接口

```
curl -G 'http://localhost:8086/query?pretty=true' --data-urlenco
de "db=testDB" --data-urlencode "q=SELECT * FROM weather ORDER B
Y time DESC LIMIT 3"
```

InfluxDB是支持类**SQL**语句的，具体的查询语法都差不多，就不再详细描述了。

数据库与表的操作

以下语句都可以直接在**InfluxDB**的**Web**管理界面中调用

```
# 创建数据库
CREATE DATABASE "db_name"
# 显示所有数据库
SHOW DATABASES
# 删除数据库
DROP DATABASE "db_name"

# 使用数据库
USE mydb

# 显示该数据库中的表
SHOW MEASUREMENTS

# 创建表
# 直接在插入数据的时候指定表名（weather就是表名）
insert weather,altitude=1000,area=北 temperature=11,humidity=-4

# 删除表
DROP MEASUREMENT "measurementName"
```

数据保存策略（Retention Policies）

InfluxDB没有提供直接删除Points的方法，但是它提供了Retention Policies。主要用于指定数据的保留时间：当数据超过了指定的时间之后，就会被删除。

查看当前数据库的Retention Policies

```
SHOW RETENTION POLICIES ON "testDB"
```

创建新的Retention Policies

```
CREATE RETENTION POLICY "rp_name" ON "db_name" DURATION 30d REPL
ICATION 1 DEFAULT
```

其中：

1. rp_name：策略名
2. db_name：具体的数据库名
3. 30d：保存30天，30天之前的数据将被删除
   它具有各种时间参数，比如：h（小时），w（星期）
4. REPLICATION 1：副本个数，这里填1就可以了
5. DEFAULT 设为默认的策略

修改Retention Policies

```
ALTER RETENTION POLICY "rp_name" ON db_name" DURATION 3w DEFAULT
```

删除Retention Policies

```
DROP RETENTION POLICY "rp_name" ON "db_name"
```

具体效果，大家可以直接自己在测试数据库上试验

连续查询（Continuous Queries）

当数据超过保存策略里指定的时间之后，就会被删除。
如果我们不想完全删除掉，比如做一个数据统计采样：把原先每秒的数据，存为每小时的数据，让数据占用的空间大大减少（以降低精度为代价）。

这就需要InfluxDB提供的：连续查询（Continuous Queries）。

当前数据库的Continuous Queries

```
# 这条命令得在命令行下输入，在web管理界面不能显示。
SHOW CONTINUOUS QUERIES
```

创建新的Continuous Queries

```
> CREATE CONTINUOUS QUERY cq_30m ON testDB BEGIN SELECT mean(temperature) INTO weather30m FROM weather GROUP BY time(30m) END
```

其中：

1. cq_30m：连续查询的名字
2. testDB：具体的数据库名
3. mean(temperature): 算平均温度
4. weather： 当前表名
5. weather30m： 存新数据的表名
6. 30m：时间间隔为30分钟

当我们插入新数据之后，可以发现数据库中多了一张名为**weather30m(**里面已经存着计算好的数据了**)**。这一切都是通过**Continuous Queries**自动完成的。

```
> SHOW MEASUREMENTS
name: measurements
------------------
name
weather
weather30m
```

删除Continuous Queries

```
DROP CONTINUOUS QUERY <cq_name> ON <database_name>
```

具体效果，大家可以直接自己在测试数据库上试验

用户管理

以下语句都可以直接在**InfluxDB**的**Web**管理界面中调用

```
# 显示用户
SHOW USERS
# 创建用户
CREATE USER "username" WITH PASSWORD 'password'
# 创建管理员权限的用户
CREATE USER "username" WITH PASSWORD 'password' WITH ALL PRIVILE
GES

# 删除用户
DROP USER "username"
```

第三方库API接口

InfluxDB提供了各种语言的Http API接口的封装。具体可以看这里：
https://docs.influxdata.com/influxdb/v0.10/clients/api/

同时，官方也提供了Telegraf插件来收集数据，除此之外还有collectd等比较常用的第三方数据收集工具。

我并不推荐一开始就用各种工具，这样会淡化对InfluxDB的理解。
当然，如果你本身对这些工具很熟悉，那么就直接使用吧！

所以本章节主要介绍各语言对InfluxDB Http API接口的封装。

NodeJS

截至2016年2月26日，官方的列表中，并没有提供NodeJS的Http API接口的封装。 - - !

好在我们有Github，在上面搜索到一个:
https://github.com/node-influx/node-influx

项目介绍写着的是支持0.9x版本的InfluxDB，我在0.10上试了下，基本可用。
因为是纯Http API接口，如果某些接口有问题的话，可以直接给他 pull request。

附上使用代码：

```
/**@type InfluxDB*/
var influx = require('influx')
var async = require("async")
var ut = require("./../../util/util.js")
var dbName = "testDB"
var tableName = "weather"
var client = influx({
    host : '192.168.0.196',
    port : 8086, // optional, default 8086
    protocol : 'http', // optional, default 'http'
    username : '',
    password : '',
    database : dbName
})
var altitudes = [1000, 5000]
var areas = ["北", "上", "广", "深"]
async.waterfall([
        function(cb){ // 创建数据库
            client.createDatabase(dbName, function(err,result){
                ut.log("createDatabase", result)
                cb(err, null)
            } )
        },
        function(result, cb){ // 获取数据库名字
            client.getDatabaseNames( function(err, result){
```

```
                ut.log("getDatabaseNames", result)
                cb(err, null)
            } )
        },
        function(result, cb){ // 写入数据
            var points = [
                [
                    {
                        temperature: ut.RandByRange(0, 100), hum
idity : ut.RandByRange(-15, 30)
                    },
                    {
                        altitude: altitudes[ut.RandByRange(0, al
titudes.length)], area : areas[0]
                    },
                ],
                [
                    {
                        temperature: ut.RandByRange(0, 100), hum
idity : ut.RandByRange(-15, 30)
                    },
                    {
                        altitude: altitudes[ut.RandByRange(0, al
titudes.length)], area : areas[1]
                    },
                ],
            ]
            client.writePoints(tableName, points, function(err,
result){
                ut.log("writePoint", result)
                cb(err, null)
            } )
        },
        function(result, cb){ // 查询数据
            client.query( 'SELECT * FROM weather ORDER BY time D
ESC LIMIT 3', function(err,result){
                ut.log("query", result)
                cb(err, null)
            } )
        },
```

```
        function(result, cb){
            client.getMeasurements( function(err,result){
                ut.log("getMeasurements", JSON.stringify(result)
)
                cb(err, null)
            })
        }
    ]
    , function(err, result){
        ut.log("finish...", err, result)
    }
)
```

数据展示工具

数据最终是需要一套UI来展示的，而这种实时数据的展示，已经有不少项目了。 比如：

1. 官方的Chronograf
2. Grafana
3. 其它...

Grafana简单使用

下载安装

Grafana也是用GO语言写的，无任何依赖，安装非常简单。

启动

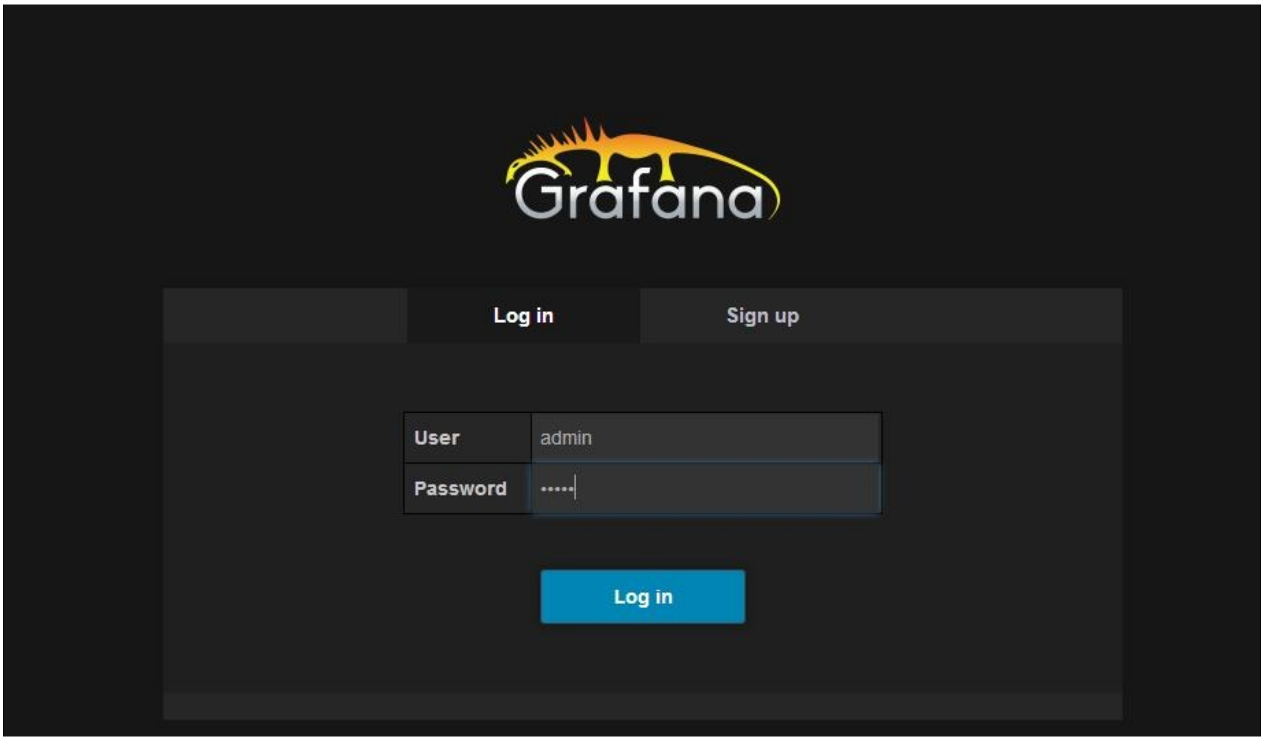
```
sudo service grafana-server start
```

运行

直接访问：http://your_ip:3000

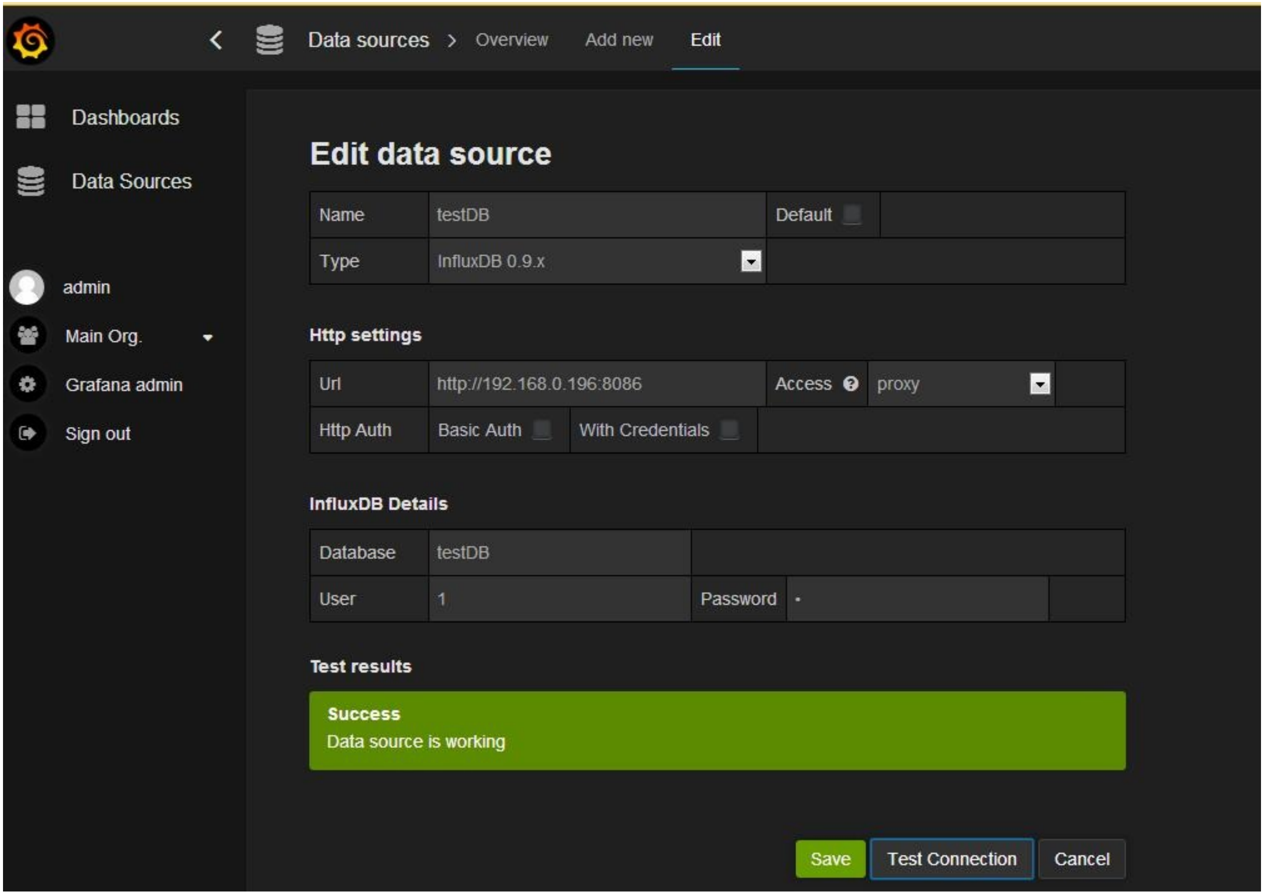
登入

默认帐号：admin
默认密码：admin



添加数据库

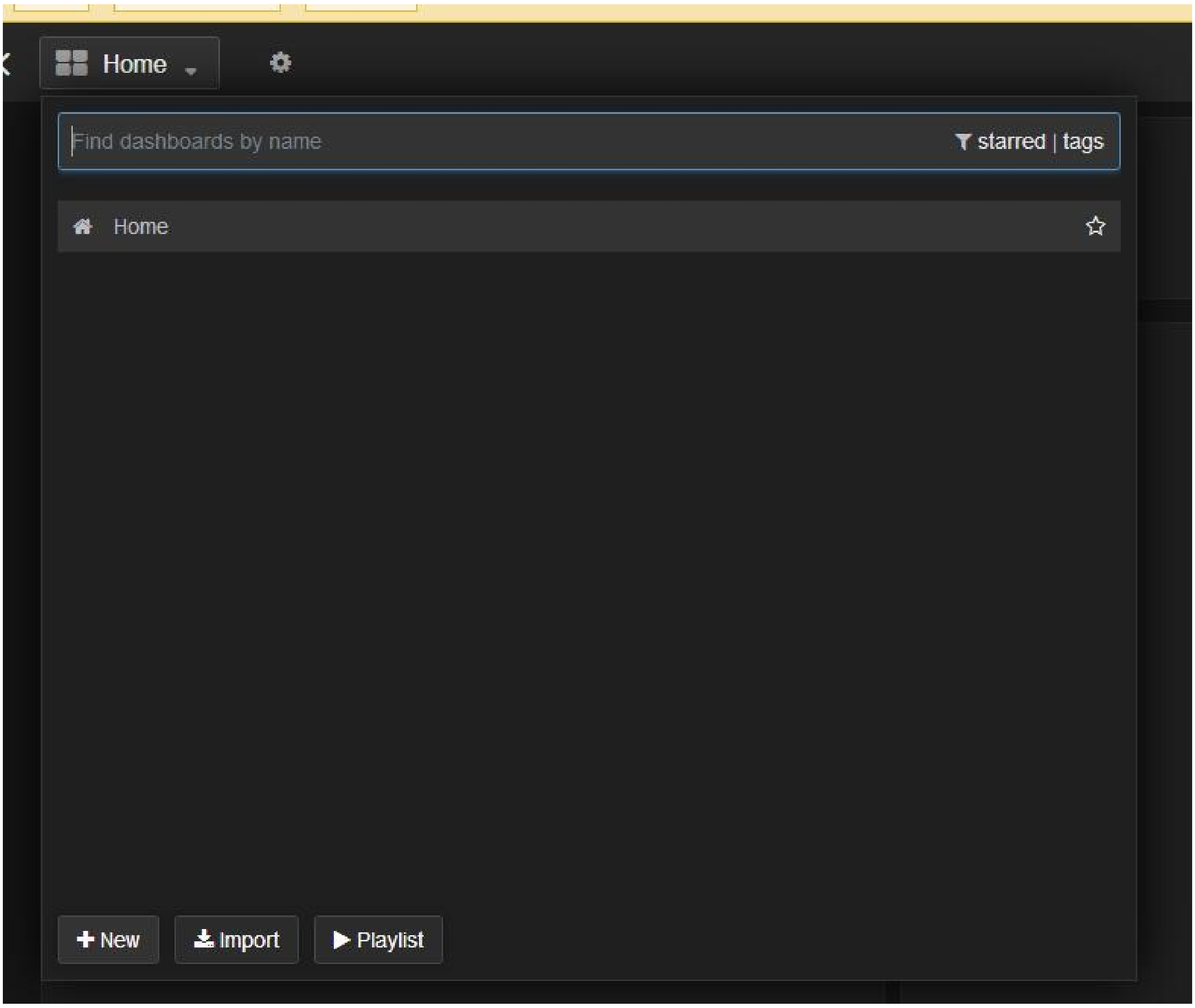
在Data Sources中添加数据库testDB



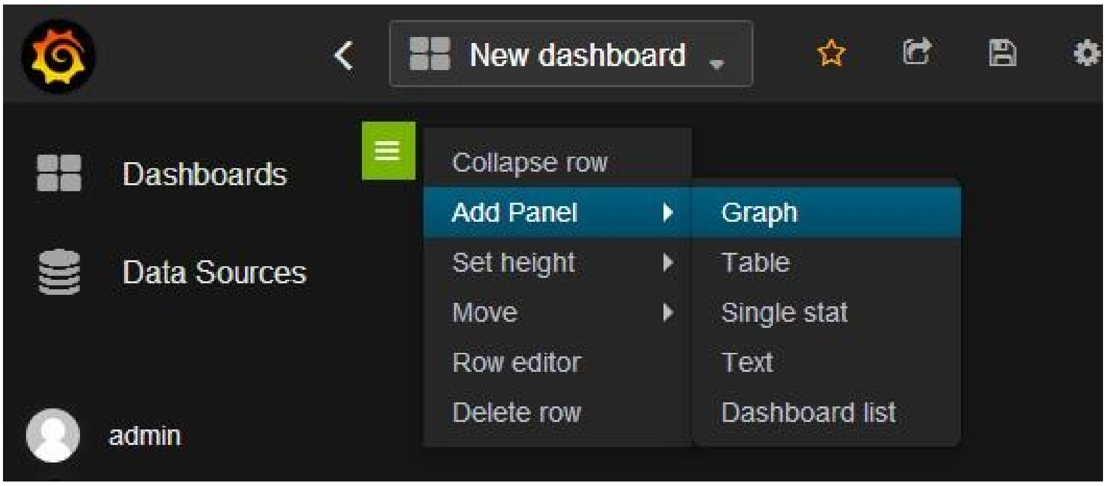
其中user和password，如果没有设置过，可以随便填下。
保存之后，可以通过Test Connection来测试，是否填写正确。

创建**Dashboard**

点击New按钮就可以了。



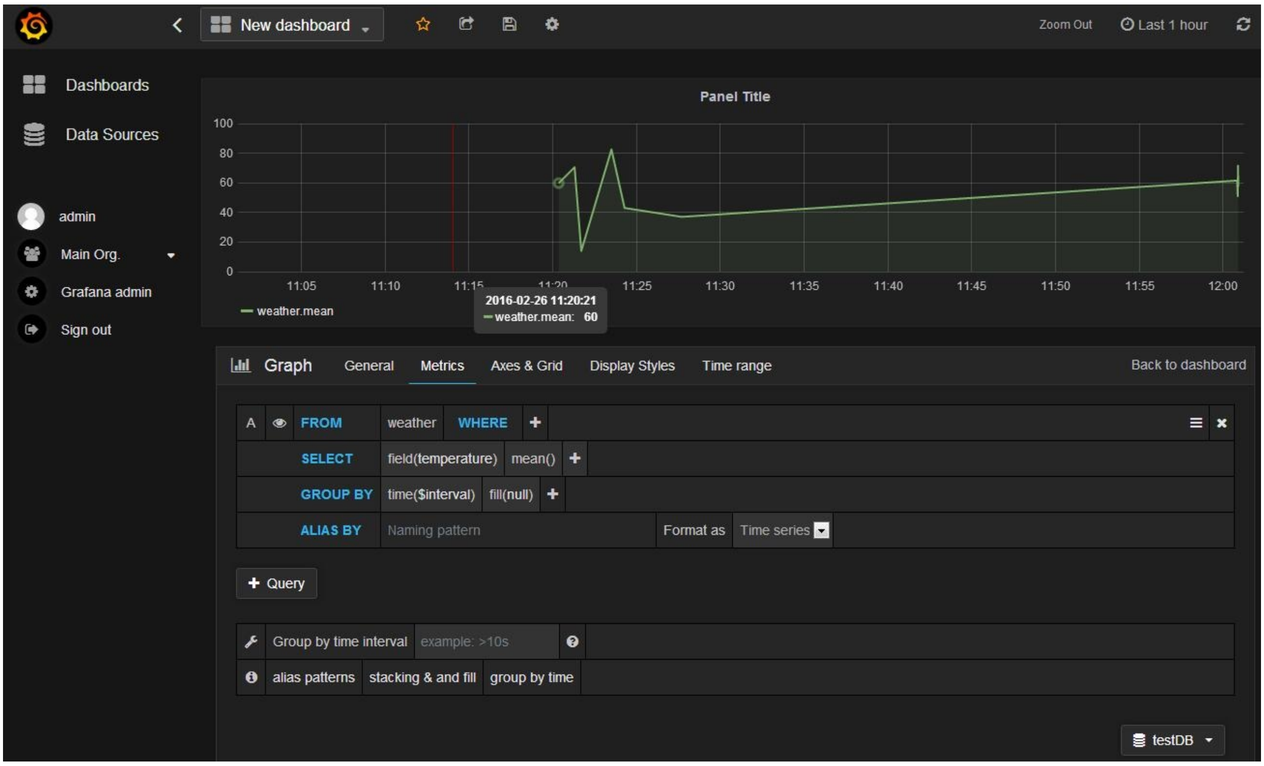
添加一个图形界面



为界面关联数据

1. 选择testDB数据库

2. 添加查询语句
3. 完美显示！记得保存



接下去可以看些什么？

1. 官方教程
2. 集群
3. 性能优化
4. 等等等...